Docket No.: 40101/08201
Ref. No.: 2000.023

REMARKS

I. INTRODUCTION

Claims 1-7 are pending in the present application. No new matter has been added. In view of the following remarks, it is respectfully submitted that all of the presently pending claims are allowable.

II. THE 35 U.S.C. § 103(a) REJECTIONS SHOULD BE WITHDRAWN

Claims 1-7 stand rejected under 35 U.S.C. § 103(a) as unpatentable over U.S. Patent No. 6,175,916 to Ginsberg et al. ("Ginsberg") in view of U.S. Patent No. 6,5429193 to Wendorf et al. ("Wendorf"). (*See 09/06/06 Office Action*, p. 2, ¶ 3).

Amended claim 1 recites, a "method, comprising: *determining a current processing mode of an executing software function*; when the current processing mode is a privileged processing mode, executing a direct program flow control instruction to directly access an instruction within a software having the privileged processing mode; and *when the current processing mode is an unprivileged processing mode*, executing an indirect program flow control instruction to cause execution of the instruction within the software having the privileged processing mode." (Emphasis added).

Ginsberg relates to a method of making a call from one process to another, wherein the method includes executing a jump instruction from a first process, while specifying an invalid destination such as an odd virtual memory address. (*See Ginsberg*, Abstract). Specifically, Ginsberg describes a system that includes a processor and a memory fault handler. (*See Id.*, col. 7, lines 4-11). The memory fault handler is described as a privileged-mode component that can change virtual memory mapping when a jump function specifies an invalid

address as the destination address. (*See Id.*). In order to change the virtual memory mappings, the processor must switch to privileged mode. (*See Id.*, col. 6, lines 63-65). If the invalid address is an odd address, then the memory fault handler recognizes that a call to a system function was attempted and executes a call processing function using the invalid address as an index into tables of available system functions. (*See Id.*, col. 7, lines 12-20). Therefore, according to the disclosure of Ginsberg, the memory fault handler, a privileged-mode component, executes a call processing function if the invalid address is an odd address. Thus, the memory fault handler executes a function that associates the invalid address with the table of available system functions. The Examiner appears to equate the performance of the memory fault handler of the Ginsberg disclosure to the executing of an indirect program flow control instruction when the current processing mode is an unprivileged processing mode, as recited in claim 1 of the present invention. (*See 09/06/06 Office Action*, p. 2, ¶ 4). However, Ginsberg explicitly states that the memory fault handler is a privileged-mode component, and in order for the memory fault handler to perform (i.e., change virtual memory mappings), the processor must switch to privileged mode. (*See Ginsberg*, col. 6 lines 63-65; col. 7, lines 4-11). Thus, the Ginsburg disclosure teaches away from the recitations in claim 1, which state, *inter alia*, "...*when the current processing mode is an unprivileged processing mode*, executing an indirect program flow control instruction to cause execution of the instruction within the software having the privileged processing mode."

Furthermore, the Examiner is correct in acknowledging that Ginsberg is silent on the claimed step of determining a current processing mode of an executing software function, recited in claim 1. (*See 09/06/06 Office Action*, p. 3, ¶ 4). However, the Examiner is incorrect in stating that Wendorf cures this deficiency. Wendorf relates to an operating system wherein the

provided protection domain support is arranged so as to be compatible with threads that obtain all their memory allocations from the operating system and were written without regard for protection domains. (*See Wendorf*, Abstract). Specifically, Wendorf relates to a process for setting a value of a protection domain register when there is a context switch (i.e., a switch from one thread to another). (*See Id.*, col. 6, line 62 – col. 7, line 17). Upon the occurrence of a context switch, a test is performed to determine if the thread whose execution is being initiated is in the protection of the operating system. (*See Id.*). The Examiner asserts that the testing of Wendorf to determine where the initiated thread is executing is equivalent to the determining a current processing mode of an executing software function, as recited in claim 1 of the present invention. (*See 09/06/06 Office Action*, p. 5, Response to Arguments). However, this assertion is incorrect. The determination of a memory location for a particular thread is not analogous to, nor equivalent to the determination of a mode of operation for a processor. The determination of a processing mode is not made on a thread-by-thread basis and the processing mode may not be dependant on the memory location for each thread. More specifically, a privileged processing mode may allow for the execution of a certain instruction, while an unprivileged processing mode may deny the execution of that very same instruction. Thus, it is respectfully submitted that Wendorf does not disclose nor suggest, "determining a current processing mode of an executing software function" as recited in claim 1.

Applicant respectfully submits that for at least the reasons stated above, claim 1 of the present application is not obvious over Ginsberg in view of Wendorf, and request that the rejection of this claim be withdrawn. As claims 2-5 depend from, and therefore include all the limitations of claim 1, it is hereby submitted that claims 2-5 are also allowable.

Claim 6 recites, *inter alia*, "identify a current processing mode of the program code segment" and "execute an indirect program flow control instruction if the current processing mode is an unprivileged processing mode." Thus, for the reasons described above with reference to claim 1, it is respectfully submitted that claim 6 is also allowable.

Claim 7 recites, *inter alia*, "identify a current processing mode of a program code segment" and "execute an indirect program flow control instruction if the current processing mode is an unprivileged processing mode." Thus, for the reasons described above with reference to claim 1, it is respectfully submitted that claim 7 is also allowable.

Docket No.: 40101/08201
Ref. No.: 2000.023

CONCLUSION

In light of the foregoing, Applicant respectfully submits that all of the now pending claims are in condition for allowance. All issues raised by the Examiner having been addressed. An early and favorable action on the merits is earnestly solicited.

Respectfully submitted,

Dated: November 6, 2006

By: _____
Michael J. Marcin (Reg. No. 48,198)

Fay Kaplun & Marcin, LLP
150 Broadway, Suite 702
New York, NY 10038